Symbolic FOND Planning for Temporally Extended Goals

Viviane Bonadia dos Santos [1], Leliane Nunes de Barros [1], Silvio do Lago Pereira [2], Maria Viviane de Menezes [3]

[1]University of São Paulo (USP)
[2] Faculdade de Tecnologia de São Paulo (FATEC-SP)
[3] Federal University of Ceará (UFC)
[1] {vbonadia, leliane}@ime.usp.br, [2] slagop@gmail.com, [3] vivianemenezes@ufc.br

Abstract

The challenges for solving a fully observable non-deterministic (FOND) planning problem involve: (i) dealing with infinite paths by generating iterative trial-and-error policies and (ii) reasoning about non conventional complex goal formulae. In this paper we propose a new algorithm to solve FOND problems with temporarily extended goals and different policy qualities. The planner, called PACTL-SYM, is based on symbolic model checking using α-CTL logic: an extension of CTL that considers actions behind the transitions. To the best of our knowledge, this is the first FOND planner that applies symbolic reasoning over the actions (and not over state transition relations) and can solve problems with complex goals. To aim this, we leverage the significant advances on symbolic reasoning in deterministic planning, the earlier works on planning as model-checking and the α-CTL temporal logic with actions, to design an efficient FOND planner based on symbolic model-checking that is sound and optimal in domains with dead-ends and cycles, allows planning for complex goals, and it is competitive with the state-of-the-art solution based on determinization and heuristic search.

Introduction

Automated planning is a long-standing field of Artificial Intelligence strongly based on a high-level description language used to define a planning task and how the agent actions may change the world. In general, planning algorithms reason over this language to automatically generate a plan of actions for a given task. Classical planning assumes the world evolves deterministically, i.e. there is no uncertainty regarding the effects of the agent actions. A more realistic situation is FOND (*Fully Observable Non-deterministic*) planning, that assumes uncertainty over the actions effects. In this case, actions can have effects that either lead the agent into a goal state or into a loop that will eventually reach the goal or a *dead-end* state (a state from which the agent can no longer achieve its goal). The objective of a FOND planning problem is to automatically synthesize (construct) a *policy*, a mapping between states and actions.

In classical planning, an R-GOAL is a *simple reachability goal* specified by a formula (a world property) possibly satisfied in a set G of terminal states. Thus, given an initial state s_0, the agent must reach s_g by executing its plan from s_0. A more complex goal is a XR-GOAL, *extended reachability goal*, that additionally specifies, externally in the form of user knowledge, a world property that must hold in all states along the plan path to s_g.

In FOND planning, due to non determinism, one can also specify what type of policy best suites the agent, i.e. if the agent will reach an R-GOAL or XR-GOAL, through: a *weak*, *strong* or *strong-cyclic* policy (Cimatti et al. 2003). Consider an R-GOAL. With a *weak policy* the agent must reach $s_g \in G$, but due to the non-determinism, it does not guarantee to do so and can possibly reach a dead-end state. With a *strong policy* the agent always achieves s_g, in spite of non-determinism. Finally, with a *strong-cyclic policy* the agent always achieves s_g, under the *fairness assumption* that execution will eventually exit from all existing cycles. While an agent with a weak policy can be seen as a *risk prone* agent (since for some tasks reaching a dead-end is unavoidable), an agent with a strong policy goal is seen as a *risk averse* agent. An agent with strong-cyclic policy goal, besides being also a *risk averse* agent, is a trial-and-error agent.

Example 1 (The Gripper Domain) *Consider an environment comprised of n rooms and a robot with two grippers. The robot can pick up or put down 1 box with its right or left gripper or 1 box with both grippers (in a safer way). The goal is to transport m boxes from an initial state to their respective target. With the deterministic action* MOVE(L1,L2)*, the robot moves from room* L1 *to room* L2*, carrying out none, one or two boxes. With the deterministic action* PUT-DOWN*, the robot put down all the boxes he is carrying on. With the action* PICKUP-R *(*PICKUP-L*), the robot picks up a box with the right (left) gripper. These actions are non-deterministic: the robot can either succeed or fail causing the box to be dropped and broken, configuring a dead-end state (none of the pick up actions can be applied in a broken box). Finally, with the non-deterministic action* PICKUP-RL *the robot uses both grippers to pick up 1 box and it can either succeed or fail (applying not enough strength to hold the box) which causes the robot to stay in the same state and creating a cycle (with no dead-ends). Note that while the robot can choose a policy of moving with 2 boxes, 1 in each gripper, it can be safer carrying out only 1 using 2 grippers*[1].

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1]Adapted from www.icaps-conference.org/competitions/

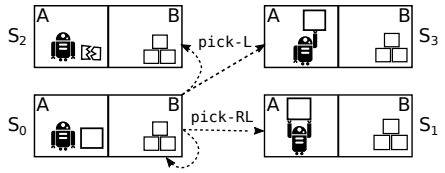


Figure 1: States of an instance in the Gripper domain with 2 rooms (A and B) and 4 boxes.

Figure 1 shows some states and actions of a small instance of Example 1 with two rooms and 4 boxes, where the goal is to have all boxes in room B. Note that the state s_2 is a dead-end. In this example, a strong-cyclic policy would repeat the action PICKUP-RL in s_0 until success and then apply action MOVE(A,B) and PUTDOWN.

In this work we focus on FOND planning tasks approaching either simple or extended reachability goals, with different plan qualities: weak, strong or strong cyclic. Furthermore, we define complex FOND goal formulae that is able to **explicitly express** a planning goal with both: the type of reachability goal and the desired policy quality. I.e. we show that the agent's risk attitude can be explicitly stated as a planning goal, and we propose algorithms to solve those complex FOND goal formulae that are based on model-checking theory and competitive with state-of-art FOND solution for R-GOALS and is the first efficient solution for XR-GOALS.

Related and previous works

Most of the solutions for FOND planning are based on heuristic search and model-checking. While heuristic search solutions can gain in efficiency, they can not give formal guarantees on the returned policy.

Examples of heuristic search to solve FOND problems are LAO* (Hansen and Zilberstein 1998), FIP (Fu et al. 2011), PRP (Muise, McIlraith, and Beck 2012) and GAMER (Kissmann and Edelkamp 2009). LAO* explicitly constructs an AND/OR graph. FIP and PRP are improvements of LAO*; they create, for every non-deterministic action, a set of deterministic actions and use an efficient deterministic planner to find a plan for each branch of the original non-deterministic action. Although PRP is considered the state-of-the-art for FOND planning, it is suboptimal w.r.t. the size of the returned policy and it can not explicitly deal with complex goals (e.g. by backtracking the search, PRP always try to find first a strong or strong cyclic policy before returning a weak policy). GAMER (Kissmann and Edelkamp 2009) is a FOND planner that translates the problem into a game against the nature that can "choose" a non-deterministic effect. It is considered a symbolic planner since it uses BDDs (Bryant 1992) in its implementation, however it performs worse than PRP and also has not been applied for tasks with complex goals.

The other main solution for FOND is based on *model checking* (MC) (Clarke and Emerson 1982), where the domain is seen as a formal system and the planning goal is a logical formula that must be satisfied in this system. Yet, the model-checking approach tries to solve a planning problem as a formal system, with guarantees over the soundness and the satisfaction of simple and complex goal formulae. Examples of model checking solutions are MBP (Cimatti et al. 2003) and PACTL (Pereira and Barros 2008b; Menezes, Barros, and Pereira 2014; Bonadia and Barros 2017). The MBP planner implements MC preimage operations using BDDs (*Binary Decision Diagrams*) (Bryant 1992) over the state transition relation $(state, action, state)$; formally, it is based on CTL (Clarke and Emerson 1982), a branching time temporal logic that does not consider actions behind the state transitions. The MBP limitations are: it does not implement CTL semantics and it uses extra-logical procedures to reason over the actions.

PACTL planner overcomes MBP limitations by using the α-CTL logic (Pereira and Barros 2008b): an CTL extension that considers the actions behind the state transitions in the CTL semantics. PACTL includes an algorithm for each type of policy goal formulae: weak, strong and strong-cyclic. In Pereira and Barros (2008b) the work is concerned with the theoretical aspects of FOND based on α-CTL and not with efficiency. In Menezes, Barros, and Pereira (2014) it was proposed the first efficient symbolic model checking algorithm based on α-CTL (VACTL-SYM), able to formally reason about actions (not about state transitions as in MBP) and, by using BDDs, it can scale up. However, VACTL-SYM can only be used to formally verify if a given policy satisfies a goal formula, not being able to synthesize policies. The symbolic version of PACTL, called PACTL-SYM (Bonadia and Barros 2017), was first proposed to synthesize weak and strong policies based on α-CTL. In Bonadia et al. (2019) it was approached the strong-cyclic policies and in Bonadia (2018), this algorithm was applied to extended reachability goals for weak, strong or strong-cyclic goal policies. This work proposes a new version of the strong-cyclic algorithm which eliminates an unnecessary loop.

Note that several works in FOND (Peot and Smith 1992; Pryor and Collins 1996; Warren 1976; Weld, Anderson, and Smith 1998) do not address the problem of dealing with strong-cyclic policies, neither complex goal formulae. With the exception of the planner SIMPLAN (Kabanza, Barbeau, and St-Denis 1997) that can deal with temporally extended goals but cannot plan for strong-cyclic policies.

A recent FOND planning work (Rodriguez et al. 2021) proposes an extended FOND model, called FOND+, a more expressive type of FOND planning model where fairness assumptions are also stated explicitly and used to explain and describe different forms of FOND planning for strong or strong-cyclic planning (not for weak planning). Although the idea of fairness comes from formal methods to express how "fair" no-deterministic events occur in a system, this paper uses this concept in an original way that integrates qualitative numerical planning (QNP) problems with different types of FOND planning problems. However, it can not deal with temporally extended goals.

Motivation

Most of the planners based on symbolic model checking (using BDDs to reason over sets of states) are specialized to deal with deterministic planning, such as the IPC (International Planning Competition) winners MIPS (Edelkamp

and Helmert 2001), SYMBA* (Torralba, López, and Borrajo 2013) and CGAMER (Torralba et al. 2017). The growing interest on symbolic approaches by the planning community is reported in Edelkamp, Kissmann, and Torralba (2015).

Table 1 summarizes the main approaches that have been used to solve deterministic and FOND planning problems. On one hand, in classical (deterministic) planning, the focus has changed from heuristic search algorithms, such as FF (Hoffmann 2001) and FAST DOWNWARD (Helmert 2006), to symbolic approaches using BDDs that reason over the actions' specification in terms of their preconditions and effects. On the other hand, in FOND planning it happened the opposite: the focus has changed from symbolic approaches, that reason over the planning problem specified in terms of state transition relations of type (s, a, s'), to the use of heuristic search like in PRP, considered the state-of-the-art. However, to the best of our knowledge, there is no FOND planner that applies formal symbolic approaches to reason over the actions' specification in some planning definition language (i.e. in terms of action's precondition and non-deterministic effects), neither that deals with complex goals.

Deterministic Planning	**FOND Planning**
1998-2012: winners of all editions of the *International Planning Competition* used **heuristic search**.	**2000-2012**: best FOND planners were based on model-checking performing **symbolic reasoning over state transition relations**.
2014-actual: top-five planners of IPC 2014 were based on MC performing **symbolic reasoning over the actions' specification.**	**2012-actual**: state-of-the-art FOND planner PRP uses **heuristic search** and determinization.

Table 1: Deterministic and FOND planning evolution.

Thus, in this work we leverage the significant advances on symbolic reasoning in deterministic planning, the earlier works on planning as model-checking and the α-CTL temporal logic with actions (Pereira and Barros 2008a) to design an efficient FOND planner based on symbolic model-checking that: (i) is sound and optimal in domains with dead-ends and cycles, (ii) allows planning for complex goals, and (iii) is competitive with the state-of-the-art solution based on determinization. To the best of our knowledge, this is the first efficient FOND planner that can solve extended reachability goals and can also synthesize weak, strong or strong-cyclic policies using symbolic CTL-based model checking (with no extra-logical proceedings). The empirical results in the analyzed domains show that PACTL-SYM outperforms the state-of-the-art FOND planner PRP based on heuristic search and MBP planner, also based on model checking.

This paper improves previous works (Bonadia et al. 2019) (Bonadia and Barros 2017) by proposing a new algorithm for strong-cyclic policies (Algorithms 1); and presenting new experiments for extended reachability goals comparing our algorithm with other FOND solutions.

FOND Planning Foundation

Fully-Observable Non-Deterministic (FOND) planning is a subclass of planning under uncertainty with fully-observable states. So, in this setting, the agent does not explicitly plan for sensing the world since the complete state description is automatically returned after the execution of an action.

Definition 1 (FOND **Planning Domain)** *Formally, a* FOND *planning domain is a tuple* $\mathcal{D} = \langle \mathcal{S}, \mathcal{L}, \mathcal{T} \rangle$ *over a set of propositional atoms* $\mathbb{P}$ *and a set of actions* $\mathbb{A}$*, such that:*

- $\mathcal{S}$ *is a finite set of states;*
- $\mathcal{L} : \mathcal{S} \mapsto 2^{\mathbb{P}}$ *is a state labeling function; and*
- $\mathcal{T} : \mathcal{S} \times \mathbb{A} \to 2^{\mathcal{S}}$ *is a non-deterministic state transition function such that, given a state* $s \in S$ *and an action* $a \in \mathbb{A}$*,* $\mathcal{T}$ *returns a set of possible next states.* ◆

A FOND planning problem can also be described by an *action description language*, such as STRIPS-like language (Fikes and Nilsson 1971) or other action languages (Pereira and Barros 2008a; Herzig, Menezes, and de Barros 2014). In this case, the set of states $\mathcal{S}$ can be induced by the actions.

Definition 2 (FOND **Planning Problem)** *A* FOND *planning problem is a tuple* $\mathcal{P} = \langle \mathcal{D}, s_0, \varphi \rangle$*, where* $\mathcal{D}$ *is a* FOND *planning domain (Def. 1),* $s_0 \in \mathcal{S}$ *is the initial state and* φ *is a logical goal formula.* ◆

A terminal state $s \in S$ is a state with no applicable actions or where the only applicable action is a self-loop action (called NO-OP). In planning, a terminal state can be a goal state or a dead-end state. Note that not all dead-end states are terminal states. A planning domain can have a *trap of dead-ends*, which is a strongly-connected component of the state space from which it is not possible to reach the goal.

For an R-GOAL, φ is a propositional formula indicating that if a state s satisfies φ, i.e. $s \vDash \varphi$, then $s = s_g$ is a terminal goal state. The set of all goal states of $\mathcal{P}$ is denoted by G. For an XR-GOAL, φ is given by a pair of propositional formulae i.e. $\varphi = (\varphi_1, \varphi_2)$, then φ is seen as a complex goal, where φ_2 defines the property that must be satisfied in the (terminal) goal state, and φ_1 defines the property that must hold in all the states visited along the path to a goal state. We call φ_2 the *target-goal* and φ_1 the *path-goal* [2].

Note that if $\varphi_1 = True$, the XR-GOAL is equivalent to an R-GOAL, i.e., $(True, \varphi_2) \equiv \varphi_2$. We call (φ_1, φ_2), where φ_1 can be equal to $True$, a *Generalized Reachability Goal* (GR-GOAL).

Definition 3 (FOND **Planning Solution)** *A solution for a* FOND *planning problem is a policy* π *that is a mapping from states to actions* $\pi : \mathcal{S} \to \mathbb{A}$*, which prescribes action* $\pi(s) \in \mathbb{A}$ *to be taken in state* $s \in S$*.* ◆

Note that, neither Def. 3 nor Def. 2 explicitly states the agent's desire for a specific policy quality, i.e. weak, strong or strong-cyclic. In the next section, we will introduce the temporal logic α-CTL and formally define φ for GR-GOALs to explicitly express the policy quality and propose a new algorithm able to synthesize policies.

[2] In this work we assume XR-GOAL where the pair (φ_1, φ_2) are propositional formulae. However we can define other types of complex goals, by relaxing this assumption.

α-CTL: a temporal logic for reasoning about actions and complex goals

α-CTL (Pereira and Barros 2008b) is a branching time temporal logic whose semantics is defined over transition-labeled Kripke structures. Figure 4a shows an example of a transition-labeled Kripke structure, where nodes are states and transitions are labeled by the actions. In spite of the importance of actions in the α-CTL's semantics, they are not used to compose α-CTL's formulae. This is because we want to impose constraints only over the states that will be visited during the policy execution, and not over the actions that will be selected during the planning task.

The formulae of α-CTL are composed by atomic propositions , logical connectives ($\neg$, $\wedge$ and $\vee$), path quantifiers ($\exists$ and $\forall$), and the following temporal operators: $\odot$ (*next*), $\boxdot$ (*invariantly*), $\diamond$ (*finally*) and $\sqcup$ (*until*). Let p be an atomic proposition. The α-CTL's syntax is inductively defined as $\varphi ::= \top \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists \odot \varphi \mid \forall \odot \varphi \mid \exists \boxdot \varphi \mid \forall \boxdot \varphi \mid \exists(\varphi \sqcup \varphi) \mid \forall(\varphi \sqcup \varphi)$.

Let $\mathbb{P}$ be a non-empty finite set of atomic propositions and $\mathbb{A}$ be a non-empty finite set of actions. An α-CTL *temporal model* over $(\mathbb{P}, \mathbb{A})$ is a transition-labeled Kripke structure $\mathcal{D} = \langle \mathcal{S}, \mathcal{L}, \mathcal{T} \rangle$, as in Def. 1. Intuitively, the α-CTL formula $\forall \odot \varphi$ holds on a state $s \in \mathcal{D}$ if and only if there *exists* an action α, whose execution in s *necessarily* leads to an immediate successor s' of s satisfying φ. The modality $\odot$ represents the set of α-*successors* of s, for *some* particular action α, and the quantifier $\forall$ requires that every state in this set can satisfy φ. The formal definition of the α-CTL's semantics is based on the concept of *preimage*.

Definition 4 (Weak Preimage in α-CTL) *Let $Y \subseteq \mathcal{S}$ be a set of states. The weak preimage of Y, denoted by $\mathcal{T}_{\exists}^{-}(Y)$ is the set $\{s \in \mathcal{S} : \exists a \in \mathbb{A}.(\mathcal{T}(s,a) \cap Y) \neq \varnothing\}$.* ◆

Definition 5 (Strong Preimage in α-CTL) *Let $Y \subseteq \mathcal{S}$ be a set of states. The strong preimage of Y, denoted by $\mathcal{T}_{\forall}^{-}(Y)$ is the set $\{s \in \mathcal{S} : \exists a \in \mathbb{A}.\varnothing \neq \mathcal{T}(s,a) \subseteq Y\}$.* ◆

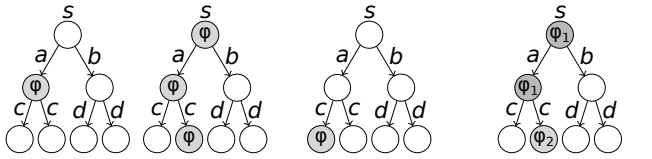
The semantics of the local temporal operators ($\exists\odot$ and $\forall\odot$) is given by preimage functions, while the semantics of the global temporal operators ($\exists\boxdot$, $\forall\boxdot$, $\exists\sqcup$ and $\forall\sqcup$) is derived from the semantics of the local temporal operators, by using least (μ) and greatest (ν) fixpoint operations.

Definition 6 (Intension of an α-CTL formula) *Let $\mathcal{D} = \langle \mathcal{S}, \mathcal{L}, \mathcal{T} \rangle$ be a temporal model (or a non-deterministic planning domain) with signature $(\mathbb{P}, \mathbb{A})$. The intension of an α-CTL formula φ in $\mathcal{D}$ (or the set of states satisfying φ in $\mathcal{D}$), denoted by $[\![\varphi]\!]_{\mathcal{D}}$, is defined as:*

- $[\![p]\!]_{\mathcal{D}} = \{s : p \in \mathcal{L}(s)\}$ (*by definition,* $[\![\top]\!]_{\mathcal{D}} = \mathcal{S}$ *and* $[\![\bot]\!]_{\mathcal{D}} = \varnothing$)
- $[\![\neg p]\!]_{\mathcal{D}} = \mathcal{S} \setminus [\![p]\!]_{\mathcal{D}}$
- $[\![(\varphi \wedge \varphi')]\!]_{\mathcal{D}} = [\![\varphi]\!]_{\mathcal{D}} \cap [\![\varphi']\!]_{\mathcal{D}}$
- $[\![(\varphi \vee \varphi')]\!]_{\mathcal{D}} = [\![\varphi]\!]_{\mathcal{D}} \cup [\![\varphi']\!]_{\mathcal{D}}$
- $[\![\exists \odot \varphi]\!]_{\mathcal{D}} = \mathcal{T}_{\exists}^{-}([\![\varphi]\!]_{\mathcal{D}})$
- $[\![\forall \odot \varphi]\!]_{\mathcal{D}} = \mathcal{T}_{\forall}^{-}([\![\varphi]\!]_{\mathcal{D}})$
- $[\![\exists \boxdot \varphi]\!]_{\mathcal{D}} = \nu Y.([\![\varphi]\!]_{\mathcal{D}} \cap \mathcal{T}_{\exists}^{-}(Y))$
- $[\![\forall \boxdot \varphi]\!]_{\mathcal{D}} = \nu Y.([\![\varphi]\!]_{\mathcal{D}} \cap \mathcal{T}_{\forall}^{-}(Y))$
- $[\![\exists(\varphi \sqcup \varphi')]\!]_{\mathcal{D}} = \mu Y.([\![\varphi']\!]_{\mathcal{D}} \cup ([\![\varphi]\!]_{\mathcal{D}} \cap \mathcal{T}_{\exists}^{-}(Y)))$
- $[\![\forall(\varphi \sqcup \varphi')]\!]_{\mathcal{D}} = \mu Y.([\![\varphi']\!]_{\mathcal{D}} \cup ([\![\varphi]\!]_{\mathcal{D}} \cap \mathcal{T}_{\forall}^{-}(Y)))$ ◆

Definition 7 (α-CTL satisfiability) *Let s be a state in a temporal model $\mathcal{D}$, and φ an α-CTL (goal) formula. The α-CTL's satisfiability relation is defined as: $(\mathcal{D}, s) \vDash \varphi \Leftrightarrow s \in [\![\varphi]\!]_{\mathcal{D}}$.* ◆

Figure 2 illustrates computational paths in models that satisfy α-CTL formulae from a state s. For example, in Figure 2c, the formula $s \vDash \exists \diamond \varphi$ expresses that exists an action whose execution in s starts *some* trajectories leading to states satisfying φ.



(a) $s \vDash \exists \odot \varphi$ (b) $s \vDash \exists \boxdot \varphi$ (c) $s \vDash \exists \diamond \varphi$ (d) $s \vDash \exists(\varphi_1 \sqcup \varphi_2)$

Figure 2: α-CTL semantics for some α-CTL operators.

In order to build a planner based on α-CTL, the definition of intension of α-CTL formulae must be reformulated so that it is possible to obtain not only the set of states that satisfy a formula but also the transitions considered during the selection of those states. Thus, the weak and strong preimages of a set of states X are redefined, respectively, as:

Definition 8 (Weak Preimage in α-CTL with transitions) *Let $X \subseteq \mathcal{S}$ be a set of states. The weak preimage of X is the set $\{(s,a) : s \in \mathcal{S}, a \in \mathcal{A}$ and $\mathcal{T}(s,a) \cap X \neq \varnothing\}$* ◆

Definition 9 (Strong Preimage in α-CTL with transitions) *Let $X \subseteq \mathcal{S}$ be a set of states. The strong preimage of X is the set $\{(s,a) : s \in \mathcal{S}, a \in \mathcal{A}$ and $\varnothing \neq \mathcal{T}(s,a) \subseteq X\}$* ◆

Specifying Complex Goals in α-CTL

As mentioned before, one advantage of planning based on model checking is to be able to express complex goals.

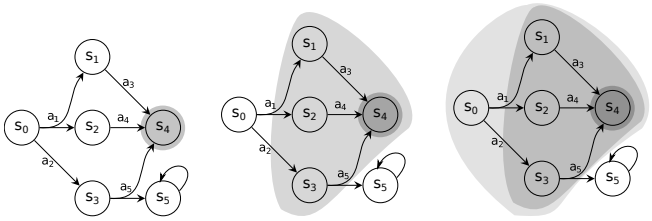
Given a FOND planning problem $\mathcal{P} = \langle \mathcal{D}, s_0, \varphi \rangle$, the complex goal φ is an α-CTL formula expressing the policy quality for generalized reachability goal (φ_1,φ_2) as follows:

- ***Weak policy for*** GR-GOAL: $\varphi = \exists(\varphi_1 \sqcup \varphi_2)$,
- ***Strong policy for*** GR-GOAL: $\varphi = \forall(\varphi_1 \sqcup \varphi_2)$, and
- ***Strong-cyclic policy for*** GR-GOAL: $\varphi = \forall \boxdot \exists(\varphi_1 \sqcup \varphi_2)$,

where $(\mathcal{D}, s_0) \vDash \varphi$; φ_2 is a propositional formula that must be satisfied in $s_g \in G$; and φ_1 is a propositional formula that must be satified in all states in the path to $s_g \in G$ for an XR-GOAL and can be equal to $\top$ for an R-GOAL.

Figure 3 presents how to compute a submodel M that satisfies $\exists \diamond \varphi_2$ (i.e. a weak policy for GR-GOAL with $\varphi_1 = \top$), supposing that $s_4 \vDash \varphi_2$. According to Def. 6 (α-CTL semantics), in order to obtain submodel M, it is necessary to compute the weak preimage, starting from the set of states $X = \{s_4\}$ (Figure 3a), until a minimal fixpoint is reached. After first iteration, the submodel computed by preimage is $M = \{s_4, (s_1, a_3), (s_2, a_4), (s_3, a_5)\}$ (Figure 3b) and, after second iteration, the submodel computed

is $M = \{s_4, (s_1, a_3), (s_2, a_4), (s_3, a_5), (s_0, a_1), (s_0, a_2)\}$ (Figure 3c), which is a minimal fixpoint since the third iteration produces this same submodel.



(a) $M = \{s_4\}$ (b) First iteration. (c) Second iteration.

Figure 3: Computing a submodel M that satisfies $\exists \diamond \varphi_2$, supposing that $s_4 \vDash \varphi_2$. Edges that start from the same state and with the same label represent the non-deterministic effects of an action. The gray shadow indicates the submodel computed by the weak preimage in α-CTL.

PACTL planner: the algorithm

Algorithm 1 shows a pseudo-code of the PACTL planner, where the main function receives a FOND problem for a generalized reachability goal with three types of quality goal: weak, strong and strong-cyclic policy quality.

Given a FOND planning problem $\mathcal{P} = \langle \mathcal{D}, s_0, \varphi \rangle$, with an XR-GOAL $\varphi = \forall(\varphi_1 \sqcup \varphi_2)$, the PACTL algorithm for **strong policies** (Algorithm 1, function PACTL-STRONG($\mathcal{P}$)) computes the set of states $G1$ that satisfies φ_1 (Line 8), path-goal state set, and the set of states $G2$, that satisfies φ_2 (Line 9), target-goal state set. Then, the algorithm calls the function MODELAU (Algorithm 1, function MODELAU($\mathcal{T}, G1, G2$)), where the set of states that satisfies $G2$ is expanded through strong preimage operations (Line 20) until reaching a fixed point ($M = M'$). Note that, at each strong preimage iteration, the state-action pairs that are not in $G1$ are pruned (Line 21). Finally, if the submodel M contains s_0, then it is possible to extract a strong policy from M (Algorithm 1, Line 11).

The PACTL algorithm for weak policies is similar to the Algorithm 1, function PACTL-STRONG($\mathcal{P}$). The main difference is that to compute a weak submodel, the algorithm uses weak preimage (Def. 4).

Given a FOND planning problem $\mathcal{P} = \langle \mathcal{D}, s_0, \varphi \rangle$, with the complex goal $\varphi = \forall \boxdot \exists(\varphi_1 \sqcup \varphi_2)$, the PACTL algorithm for **strong-cyclic policies** (Algorithm 1, function PACTL-STRONGCYCLIC($\mathcal{P}$)) starts by computing the sets G1 and G2, in Lines 26 and 27; Then it adds for each target-goal state s_g a self-loop action NO-OP, called τ (Line 28). The algorithm is based on two functions:

- MODELEU($\mathcal{T}, G1, G2$): computes (*synthesizes*) a submodel M as a set of state-action pairs that satisfies $\exists(\varphi_1 \sqcup \varphi_2)$ by computing the **weak preimage** (Def. 8), until a fixpoint is reached. Whenever a weak preimage is computed, the state-action pairs that are not in $G1$ are pruned, that is, the set obtained after each preimage step is $X = \{(s, a) \in \text{WEAKPREIMAGE}(X) : s \in G1\}$. An important proceeding to do when sinthesizing this model is to prune self-loop actions for non goal states (i.e., self-loop actions that are different of τ), since they will never be selected in the final policy; and
- MODELAG($\mathcal{T}, S$): receives the set of states S, containing all and only the states that belong to the submodel M (synthesized by MODELEU), and synthesizes a new submodel M as a set of state-action pairs that necessarily reach some state in S. To synthesize the new M, MODELAG starts from the set S and applies the **strong preimage** operation (Def. 9), until a fixpoint is reached.

Figure 4 illustrates the execution of PACTL-STRONGCYCLIC to solve a FOND planning problem with the path-goal $\varphi_1 = \top$, where s_0 is the initial state and s_g is a goal state (i.e. it satisfies φ_2). Figure 4b show the submodel with the new transition added (Algorithm 1, Line 28), Figure 4c shows the submodel M returned after the call to MODELEU (Line 29), resulting in $M = \{(s_0, a_1), (s_0, a_2), (s_1, a_3), (s_2, a_4), (s_g, \tau)\}$. Then, M is contracted by extracting from it a strong-cyclic submodel M (Figure 4d with a call to MODELAG (Line 31)), resulting in $M = \{(s_0, a_1), (s_1, a_3), (s_g, \tau)\}$. Note that MODELEU computes a submodel M that can contain a weak policy if $s_0 \in M$.

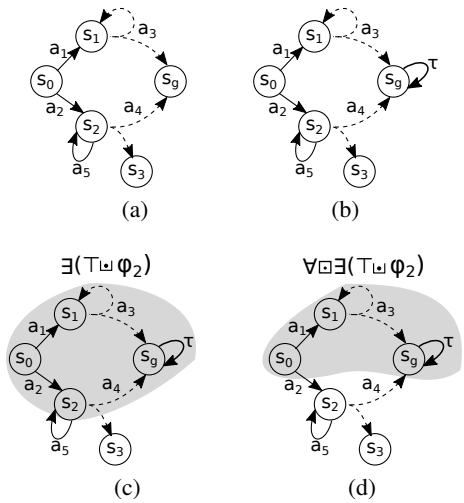


Figure 4: PACTL-STRONGCYCLIC execution; gray shadow is the submodel returned by MODELEU and MODELAG.

PACTL-SYM planner: the symbolic algorithm

The PACTL is correct since it is based on formal model checking theory. However, it is not efficient, since like MBP it works with the tuples (s, a, s'). This section presents the PACTL-SYM planner, a symbolic version of PACTL that uses efficient operations based on symbolic model checking by reasoning over action descriptions. We first define how to represent states and actions as logical formulae and the main symbolic operations. Then we introduce BDDs, a fundamental data structure for symbolic reasoning used in the implementation of our proposed planner.

Algorithm 1: PACTL($\mathcal{P}$)

Function `Main(`$\mathcal{P}$`)`**:**
  Input : $\mathcal{P} = \langle \mathcal{D}, s_0, \varphi \rangle$ where $\mathcal{D} = \langle \mathcal{S}, \mathcal{L}, \mathcal{T} \rangle$
  Output: A policy or FAIL
  switch φ **do**
    case $\exists(\varphi_1 \sqcup \varphi_2)$ **do return** PACTL-WEAK($\mathcal{P}$)
    case $\forall(\varphi_1 \sqcup \varphi_2)$ **do return** PACTL-STRONG($\mathcal{P}$)
    case $\forall \diamond \exists(\varphi_1 \sqcup \varphi_2)$ **do return** PACTL-STRONGCYCLIC($\mathcal{P}$)

Function PACTL-STRONG($\mathcal{P}$)**:**
  Input : $\mathcal{P} = \langle \mathcal{D}, s_0, \varphi \rangle$ where $\mathcal{D} = \langle \mathcal{S}, \mathcal{L}, \mathcal{T} \rangle$, $\varphi = \forall(\varphi_1 \sqcup \varphi_2)$.
  Output: A policy or FAIL
  $G1 \leftarrow \{s \in \mathcal{S} \mid s \vDash \varphi_1\}$ `// set of path-goal states; G1 = ∅ if R-GOAL`
  $G2 \leftarrow \{s \in \mathcal{S} \mid s \vDash \varphi_2\}$ `// set of target-goal states`
  $M \leftarrow$ MODELAU($\mathcal{T}, G1, G2$)
  if $s_o \in M$ **then return** EXTRACTPOLICY(M)
  else return FAIL

Function MODELAU($\mathcal{T}, G1, G2$)**:**
  Input : $G1$: path-goal states, $G2$: target-goal states and $\mathcal{T}$: state transition function
  Output: Submodel M
  $M \leftarrow G2$
  $M' \leftarrow \varnothing$
  while $M \neq M'$ **do**
    $M' \leftarrow M$
    $C \leftarrow$ STATES(M) `// submodel states`
    $P \leftarrow$ STRONGPREIMAGE(C)
    $P' \leftarrow \{(s, a) \in P$ and $s \in G1\}$
    $M \leftarrow M \cup P'$
  return M

Function PACTL-STRONGCYCLIC($\mathcal{P}$)**:**
  Input : $\mathcal{P} = \langle \mathcal{D}, s_0, \varphi \rangle$ where $\mathcal{D} = \langle \mathcal{S}, \mathcal{L}, \mathcal{T} \rangle$, $\varphi = \forall \boxdot \exists(\varphi_1 \sqcup \varphi_2)$ and φ_1 and φ_2 are propositional formulae.
  Output: A policy or FAIL
  $G1 \leftarrow \{s \in \mathcal{S} \mid s \vDash \varphi_1\}$
  $G2 \leftarrow \{s \in \mathcal{S} \mid s \vDash \varphi_2\}$
  `// add τ-transitions on G2 states`
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{t = (s, a, s) \mid s \in G2$ and $a = \tau = noop\}$
  $M \leftarrow$ MODELEU($\mathcal{T}, G1, G2$)
  $S \leftarrow$ STATES(M) `// submodel states`
  $M \leftarrow$ MODELAG($\mathcal{T}, S$)
  if $s_o \in M$ **then return** EXTRACTPOLICY(M)
  else return FAIL

Function MODELEU($\mathcal{T}, G1, G2$)**:**
  Input : $G1$: path-goal states, $G2$: target-goal states and $\mathcal{T}$: state transition function
  Output: Submodel M
  $M \leftarrow G2$
  $M' \leftarrow \varnothing$
  while $M \neq M'$ **do**
    $M' \leftarrow M$
    $C \leftarrow$ STATES(M) `// submodel states`
    $P \leftarrow$ WEAKPREIMAGE(C)
    $P' \leftarrow \{(s, a) \in P$ and $s \in G1\}$
    $P'' \leftarrow P' \setminus \{(s, a) \in P', (s, a, s) \in \mathcal{T}$ and $a \neq \tau\}$
    $M \leftarrow M \cup P''$
  return M

Symbolic Representation of States and Actions

Symbolic approaches for planning is a trending research area in artificial intelligence (Edelkamp, Kissmann, and Torralba 2015; Torralba et al. 2017). In order to propose a symbolic planning method, we need to represent the planning problem in terms of logical formulae. Consider a planning domain $\mathcal{D}$ defined over a set of atomic propositions $\mathbb{P}$ and a set of actions $\mathbb{A}$. Its logical representation, denoted by $\mathcal{D}_{sym}$, involves representing a state s as a formula:

$$\xi(s) = \bigwedge_{p \in \mathcal{L}(s)} p \wedge \bigwedge_{q \in \mathbb{P} \setminus \mathcal{L}(s)} \neg q, \tag{1}$$

and a set of states $X \subseteq \mathcal{S}$ can be represented by a disjunction of every state $s \in X$, as showed bellow:

$$\xi(X) = \bigvee_{s \in X} \xi(s). \tag{2}$$

The symbolic MC done by MBP (Bertoli et al. 2001) uses formulae representing the planning model as a conjunction $\xi(s) \wedge a \wedge \xi(s')$. However, we need a better model representation, one that allows to generate states on demand that are based on action description, as follows.

Another way to represent the planning model is to use a STRIPS based language (Fikes and Nilsson 1971), i.e. in terms of actions preconditions and effects. A precondition of action a, denoted by $prec(a)$, defines a set of atomic propositions that must be true in a state s where the action will be executed, and the effects represent the changes in s after the execution of a. The positive effects of action a, denoted by $\mathit{eff}^+(a)$, are the set of atomic propositions that become true in s and the negative effects, denoted by $\mathit{eff}^-(a)$, are the set of atomic propositions that become false in s. When actions are non-deterministic, $\mathit{eff}(a)$ represents a set of effects i.e. $\mathit{eff}(a) = \{e_1, e_2, \cdots, e_n\}$ where each $e_i \in \mathit{eff}(a)$ is given by $\mathit{eff}^+(a, e_i)$ and $\mathit{eff}^-(a, e_i)$. Thus, a non-deterministic action $a = \langle prec(a), \mathit{eff}(a, e_1), \cdots, \mathit{eff}(a, e_n) \rangle$ is represented by the following propositional formulae:

$$\xi(prec(a)) = \bigwedge_{p \in prec(a)} p \qquad \text{and} \tag{3}$$

$$\xi(\mathit{eff}(a, e_i)) = \Big(\bigwedge_{q \in \mathit{eff}^+(a, e_i)} q \wedge \bigwedge_{r \in \mathit{eff}^-(a, e_i)} \neg r \Big), e_i \in \mathit{eff}(a). \tag{4}$$

Moreover, we include for each non-deterministic effect $e_i \in \mathit{eff}(a)$, the set $\mathit{changes}(a, e_i)$ of all atomic propositions occurring in e_i. Below, see an example of a symbolic representation of an action a in a domain with $\mathbb{P} = \{p, q, r\}$.

$\xi(\textbf{\textit{prec}}(a)) : p \wedge q$
$\xi(\textbf{\textit{eff}}(a, e_1)) : \neg q \wedge \neg r, \textbf{\textit{changes}}(a, e_1) = \{q, r\}$
$\xi(\textbf{\textit{eff}}(a, e_2)) : \neg p \wedge \neg r, \textbf{\textit{changes}}(a, e_2) = \{p, r\}$

As mentioned before, a FOND planning problem can be described by an action description language. In this context, the planning domain is given by the set of actions $\mathcal{A}$ and the set of states are induced by the actions. Thus, a symbolic FOND planning problem is $\mathcal{P}_{Sym} = \langle \mathcal{A}, \xi(s_0), \varphi \rangle$ where $\mathcal{A}$ is the set of actions with each action $a \in \mathcal{A}$ represented by the tuple $\langle \xi(prec(a)), \xi(\mathit{eff}(a, e_1)), \mathit{changes}(a, e_1), \cdots, \xi(\mathit{eff}(a, e_n)), \mathit{changes}(a, e_n) \rangle$; $\xi(s_0)$ is a formula representing the initial state and φ is a

formula representing a complex goal. The main advantage of this representation is to be able to build the state space on demand by applying actions from the initial state (progressive search) or from the set of states satisfying φ (regressive search), to generate the successor or ancestor states, respectively.

Progression and Regression Symbolic Operations

Given a planning problem $\mathcal{P}_{Sym} = \langle \mathcal{A}, \xi(s_0), \varphi \rangle$ we can generate the complete state-space through the progression and regression operations. To define these operations we need a logic that extends the propositional logic with quantifiers over the proposition values, called QBF logic (*Quantified Boolean Formulae*) (Büning and Bubeck 2009). Given a propositional formula φ and an atomic proposition p occurring in φ, the existential quantifier in QBF is defined as $\exists p.\varphi \equiv \varphi[\top/p] \vee \varphi[\bot/p]$. Similarly, the universal quantifier is defined as $\forall p.\varphi \equiv \varphi[\top/p] \wedge \varphi[\bot/p]$. Quantifiers can also be defined for a set of atomic propositions. Let $P = \{p_1, p_2, \cdots, p_n\}$ be a subset of atomic propositions occurring in φ. We define $\exists P.\varphi$ as $\exists p_1.(\cdots \exists p_n.\varphi)$; and $\forall P.\varphi$ as $\forall p_1.(\cdots \forall p_n.\varphi)$.

Given a set of states X and an action a, the **progresssion** of X by an action a returns a set of states Y that is achieved when a is executed in states of X. Given a set of states X and an action a, the **regression** computes the predecessors of X by a, i.e., a set Y of abstract states from which the execution of a results in states of X. Intuitively, a regression is computed by verifying: (1) if action a is relevant for X (i.e. if its positive effects are satisfied in X and its negative are not); (2) if positive effects of action a are eliminated from X; and (3) if the precondition of action a are added in X.

The symbolic progression and regression operations for deterministic actions were proposed by Fourman (2000). In Menezes, Barros, and Pereira (2014) these operations were extended for non-deterministic actions (Definitions 10, 11 and 12).

Definition 10 (Symbolic Progression) *The progression of a subset of states X by a non-deterministic action a computes the set of states that can be reached when a is executed in some state $s \in X$. Formally, given a subset of states $\xi(X)$ and a symbolic action $a = \langle \xi(prec(a)), \xi(\mathit{eff}(a, e_1)), \mathit{changes}(a, e_1), \cdots, \xi(\mathit{eff}(a, e_n)), \mathit{changes}(a, e_n) \rangle$, the symbolic progression, denoted as* **PROGRESSION**$(\xi(X), a)$*, is:*

$$\Big(\bigvee_{e_i \in \{1 \ldots n\}} \exists \mathit{changes}(a, e_i).(\xi(X) \wedge \xi(prec(a))) \Big) \wedge \xi(\mathit{eff}(a, e_i)). \tag{5}$$

Definition 11 (Symbolic Weak Regression) *The weak regression of a subset of states X by a non-deterministic action a, computes the set of states from which an state in X is reached by some effect of a. Thus, given a set of states $\xi(X)$ and a symbolic action a, the symbolic weak regression, denoted by* **WEAKREGRESSION** $(\xi(X), a)$*, is:*

$$\xi(prec(a)) \wedge \Big(\bigvee_{e_i \in \{1 \ldots n\}} \exists \mathit{changes}(a, e_i).(\xi(\mathit{eff}(a, e_i)) \wedge \xi(X)) \Big). \tag{6}$$

Definition 12 (Symbolic Strong Regression) *The strong regression of a set of states X computes the states from which all the non-deterministic effects of a reach a state in X. Thus, given a formula $\xi(X)$ representing the set of states X, we can compute the symbolic strong regression of a, denoted by* **STRONGREGRESSION**$(\xi(X), a)$*, as:*

$$\xi(prec(a)) \wedge \Big(\bigwedge_{e_i \in \{1 \ldots n\}} \exists \mathit{changes}(a, e_i).(\xi(\mathit{eff}(a, e_i)) \wedge \xi(X)) \Big). \tag{7}$$

It is possible to prove that the above regression operations are equivalent to the corresponding preimage functions (Rintanen 2008). However, we claim that regression operations are more appropriate for planning than preimage, since the compilation of the actions specification into the transition relation $\mathcal{T}$ leads to a combinatorial explosion triples (s, a, s').

Binary Decision Diagrams (BDDs) (Bryant 1992) are a canonical representation for boolean functions $f : \{0,1\}^n \rightarrow \{0,1\}$. A BDD is similar to a binary decision tree: an acyclic graph where non-terminal nodes (decision nodes) are labelled with boolean variables and terminal nodes are labelled with 0 or 1. For compact representation, BDDs remove duplicate terminal nodes and redundant tests. They also impose a variable ordering to allow efficient operations over boolean functions (or propositional formulae) which have exponential size in other representations, such as truth tables and conjunctive normal forms.

PACTL-SYM, implements the Algorithm 1 using BDDs to encode states and actions and to perform efficient symbolic model-checking operations, i.e. to perform the symbolic reasoning over states and actions. Furthermore, to extract a policy, we constructed a layered structure based on the work of Fourman (2000).

Empirical Analysis

PACTL-SYM was implemented in C++ using the CUDD BDD library. In thi section, we compare our planner with PRP (using the code available in https://github.com/QuMuLab/planner-for-relevant-policies), which is considered the state-of-the-art FOND planner, and with MBP (available in http://mbp.fbk.eu/), that is also based on model-checking. We execute experiments considering R-GOAL and XR-GOAL and we compare the computation time of the analysed planners in two benchmark planning domains: the Gripper and the Triangle Tireworld domains.

Experiments in the Gripper domain

We solve problems in the Gripper domain (Example 1), varying the number of boxes on the floor. The R-GOAL (φ_2) is to *move all boxes to a target location*, i.e. $\forall \boxdot \exists (\top \sqcup \varphi_2)$. The XR-GOAL has a path-goal formula (φ_1) specifying that *either both grippers must carry out the same box or both grippers are free*, which forces the robot not selecting the actions PICKUP-R and PICKUP-L, avoiding a dead-end state, i.e., $\forall \boxdot \exists (\varphi_1 \sqcup \varphi_2)$.

Table 2 shows the time (sec) spent by the planners to return a strong-cyclic policy for a complex goal formula. Considering first the results for the R-GOAL (third, fourth and fifth columns), we observe that PACTL-SYM-STRONGCYCLIC presented the best results for all analysed instances. When comparing PACTL-SYM with MBP, our planner was 5 orders of magnitude faster than MBP for the Gripper-5 instance. Although based on symbolic model checking, MBP were not able to solve instances larger than Gripper-5; This is because it uses preimage operations over state transitions (s, a, s'), which is time and memory consuming when compared with the symbolic operations used by our planner. When comparing PACTL-SYM with PRP, we have better time values for the experiments with R-GOAL (3rd column). One reason for the PRP worse performance, is that heuristics for this domain were not able to easily find deterministic plans that are part of a strong-cyclic policy.

Considering now the results for the XR-GOAL (Table 2, 2nd column), PACTL-SYM presented an even better result when compared with PRP. The reason for this is that the path-goal formula φ_1 makes PACTL-SYM **to avoid dead-ends** and therefore we can describe the complex goal with a **weak policy for XR-GOAL**, instead of a **strong-cyclic policy for XR-GOAL**. This is because with no dead-ends, a policy returned by PACTL-SYM-WEAK is in fact a strong-cyclic policy, which explains the better results.

	PACTL-SYM		PRP	MBP
	XR-GOAL	R-GOAL	R-GOAL	R-GOAL
Gripper-1	0	0	0	0
Gripper-2	0	0	28.47	0.07
Gripper-3	0	0.01	28.85	0.99
Gripper-4	0	0.01	29.12	681.32
Gripper-5	0	0.02	29.83	1362.68
Gripper-6	0.01	0.04	30.14	-
Gripper-7	0.01	0.07	31.59	-
Gripper-8	0.02	0.11	30.73	-
Gripper-9	0.02	0.19	31.56	-
Gripper-10	0.03	0.29	32.29	-
Gripper-11	0.04	0.43	33.64	-
Gripper-12	0.06	0.63	33.19	-
Gripper-13	0.08	0.93	34.06	-
Gripper-14	0.11	1.29	34.79	-
Gripper-15	0.14	1.78	34.85	-
Gripper-16	0.18	2.69	35.21	-
Gripper-17	0.23	4.61	37.71	-
Gripper-18	0.28	6.69	36.62	-
Gripper-19	0.34	9.65	38.08	-
Gripper-20	0.41	10.11	37.91	-

Table 2: Time (sec) to compute a strong-cyclic policy.

To evaluate PACTL-SYM-STRONG, we changed action PICKUP-R to be deterministic, otherwise there was no strong solutions. Still, the action PICKUP-L can introduce dead-ends and action PICKUP-RL, cycles. Table 3 shows the results of PACTL-SYM with a strong and R-GOAL formula, outperforming PRP and MBP.

	PACTL-SYM	PRP	MBP
Instances	R-GOAL	R-GOAL	R-GOAL
Gripper-1	0	0	0
Gripper-2	0	12.88	0.07
Gripper-3	0	13.24	0.79
Gripper-4	0.01	12.80	14.34
Gripper-5	0.01	13.54	619.65
Gripper-10	0.16	15.06	-
Gripper-15	0.72	16.82	-
Gripper-20	5.21	17.84	-

Table 3: Time (sec) to compute a strong policy.

Experiments in the Triangle Tireworld domain

In this domain, a vehicle can move through one-way routes. The objective is to go from an initial to a target location. In each movement, there is a non-deterministic effect of puncturing a tire. States with a flat-tire and no spare-tire are dead-ends. There is a non-deterministic action MOVE(L1, L2) and a deterministic action CHANGE-TIRE(L), applicable if the location L has a spare-tire. The topology of the locations and routes configures a triangle whose short path passes through locations with no spare-tires while the longest path has infinity spare-tires in each location. Instances vary in terms of locations number and locations with spare-tires. The R-GOAL formula is *to reach the target location* and the XR-GOAL has the path-goal formula specifying *the vehicle should pass only through locations with spare-tire* (to avoid dead-ends possibly caused by having a flat-tire).

Since heuristics to estimate distance are very efficient, PRP presented better times compared to PACTL-SYM when solving instances with R-GOAL (3^{rd} and 4^{th} columns of Table 4). However, when solving the same instances with XR-GOAL that **avoids dead-ends**, PACTL-SYM outperforms PRP (2^{nd} column of Table 4). Since in PRP we can not express XR-GOALs, nor solving problems with this type of complex goal, this experiment shows the great advantage of PACTL-SYM over PRP: the capability to express planning problems with XR-GOAL that can use knowledge about dead-ends or any constraints to help planning.

	PACTL-SYM		PRP
Instances	XR-GOAL	R-GOAL	R-GOAL
TriangleTire_1	0	0	0.02
TriangleTire_2	0	0.01	0.05
TriangleTire_3	0	0.05	0.09
TriangleTire_4	0.01	0.41	0.11
TriangleTire_5	0.01	8.55	0.15
TriangleTire_6	0.03	-	0.21
TriangleTire_7	0.05	-	0.23
TriangleTire_8	0.07	-	0.28
TriangleTire_9	0.13	-	0.31
TriangleTire_10	0.24	-	0.37

Table 4: Time (sec) to compute a strong-cyclic policy.

Conclusions

We have proposed the first FOND planner, called PACTL-SYM, that can solve problems with complex goals that explicitly describe both, the type of reachability goal (simple or extended) and the desired plan quality (weak, strong or strong-cyclic). PACTL-SYM is competitive in time with PRP, considered the state-of-the-art planner for FOND, and can be even better in problems with cycles and dead-ends of difficult detection (i.e. that can not be easily detected by its heuristics). For problems with R-GOAL where PRP can be better than PACTL-SYM (Table 4), we show that PACTL-SYM can outperform PRP when **we solved the same instances but with an XR-GOAL to express the knowledge about dead-ends for these problems** (which seems to compensate the heuristics used by PRP).

In fact, PACTL-SYM is the first FOND planner to efficiently solve complex planning goals, such as R-GOAL for weak, strong or strong-cyclic policies, expressed in α-CTL. Note that since PRP **can not express or solve FOND problems with XR-GOAL**, this comparison is able to show an important advantage of PACTL-SYM over PRP. The efficiency of PACTL-SYM comes from its symbolic reasoning over the actions, and not over the state transition relation, as done by MBP, that is also a symbolic model-checking based planner. Empirical results show that, in the Gripper Domain, PACTL-SYM is better in time than PRP with up to 3 orders of magnitude. And in the Triangle Tireworld domain, PACTL-SYM outperforms PRP when solving the corresponding instances with XR-GOAL, avoiding the dead-ends of this domain.

Acknowledgement We thank FAPESP (19/07665-4), CNPq (148935/2021-4) and FUNCAP (04772314/2020) for the financial support.

References

Bertoli, P.; Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2001. MBP: a model based planner. In *Proc. of the IJCAI'01 - Workshop on Planning under Uncertainty and Incomplete Information*.

Bonadia, V.; Barros, L.; ; and Menezes, M. V. 2019. Symbolic Planning for Strong-Cyclic Policies. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, 168–173. IEEE.

Bonadia, V.; and Barros, L. N. 2017. PACTL-SYM: um planejador baseado em verificação simbólica de modelos. *XIV Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*.

Bonadia, V. d. S. 2018. *Planejamento Baseado em Verificação Simbólica de Modelos*. Ph.D. thesis, University of São Paulo.

Bryant, R. E. 1992. Symbolic Boolean Manipulation with Ordered Binary-decision Diagrams. *ACM Comput. Surv.*, 24: 293–318.

Büning, H. K.; and Bubeck, U. 2009. Theory of Quantified Boolean Formulas. In *Handbook of Satisfiability*, 735–760. IOS Press.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, Strong, and Strong Cyclic Planning via Symbolic Model Checking. *Artificial Intelligence Journal*, 147: 35–84.

Clarke, E. M.; and Emerson, A. 1982. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *Logic of Programs*, 52–71.

Edelkamp, S.; and Helmert, M. 2001. MIPS: The Model-Checking Integrated Planning System. *AI Magazine*, 22: 67–72.

Edelkamp, S.; Kissmann, P.; and Torralba, A. 2015. BDDs strike back (in AI planning). In *Proc. of AAAI*.

Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. In *IJCAI*, 608–620.

Fourman, M. 2000. Propositional Planning. In *Proc. Workshop on Model Theoretic Approaches to Planning (AIPS)*, 10–17.

Fu, J.; Ng, V.; Bastani, F.; and Yen, I.-L. 2011. Simple and fast strong cyclic planning for fully-observable nondeterministic problems. In *IJCAI*.

Hansen, E. A.; and Zilberstein, S. 1998. Heuristic Search in Cyclic AND/OR Graphs. In *Proc. of AAAI*, 412–418.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research (JAIR)*, 26(1): 191–246.

Herzig, A.; Menezes, M. V.; and de Barros, L. N. 2014. On the revision of planning tasks. In *21st European Conference on Artificial Intelligence (ECAI)*.

Hoffmann, J. 2001. FF: The Fast-Forward Planning System. *AI magazine*, 22: 57–62.

Kabanza, F.; Barbeau, M.; and St-Denis, R. 1997. Planning control rules for reactive agents. *Artificial Intelligence*, 95(1): 67–113.

Kissmann, P.; and Edelkamp, S. 2009. Solving fully-observable non-deterministic planning problems via translation into a general game. In *German Conference on Advances in AI*, 1–8. Springer.

Menezes, M. V.; Barros, L. N.; and Pereira, S. L. 2014. Symbolic Regression for Non-Deterministic Actions. *Learning and Nonlinear Models Journal*, 12: 98–114.

Muise, C. J.; McIlraith, S. A.; and Beck, J. C. 2012. Improved Non-Deterministic Planning by Exploiting State Relevance. In *ICAPS*.

Peot, M. A.; and Smith, D. E. 1992. Conditional nonlinear planning. In *Artificial Intelligence Planning Systems (AIPS) International Conference*, 189–197.

Pereira, S. d. L.; and Barros, L. N. 2008a. Using α-ctl to Specify Complex Planning Goals. In *Logic, Language, Information and Computation*, 260–271. Springer.

Pereira, S. L.; and Barros, L. N. 2008b. A logic-based agent that plans for extended reachability goals. *AAMAS Journal*, 16: 327–344.

Pryor, L.; and Collins, G. 1996. Planning for contingencies: A decision-based approach. *Journal of Artificial Intelligence Research*, 4: 287–339.

Rintanen, J. 2008. Regression for Classical and Nondeterministic Planning. In *ECAI 2008*, 568–572.

Rodriguez, I. D.; Bonet, B.; Sardiña, S.; and Geffner, H. 2021. Flexible FOND planning with explicit fairness assumptions. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 31, 290–298.

Torralba, A.; Alcázar, V.; Kissmann, P.; and Edelkamp, S. 2017. Efficient Symbolic Search for Cost-optimal Planning. *AIJ*, 242.

Torralba, A.; López, C. L.; and Borrajo, D. 2013. Symbolic merge-and-shrink for cost-optimal planning. In *IJCAI*.

Warren, D. H. 1976. Generating conditional plans and programs. In *Proceedings of the 2nd Summer Conference on Artificial Intelligence and Simulation of Behaviour*, 344–354.

Weld, D. S.; Anderson, C. R.; and Smith, D. E. 1998. Extending graphplan to handle uncertainty & sensing actions. In *AAAI/IAAI*, 897–904.